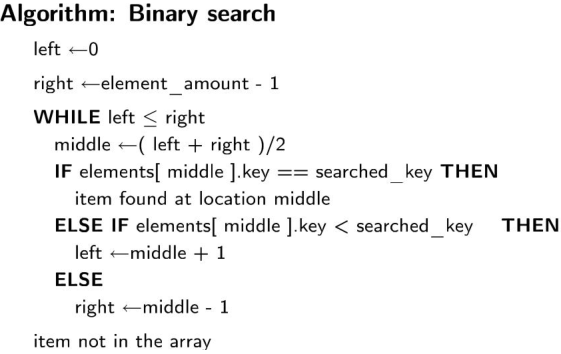
**EXERCISE # 10**

**Name: Qian Yibin Student No.: 232807**

**Exercise 1**

a)

b) 0 1 2 3 4 5 6 7 8 9 10 11. the best case: search 5; the worst case: search 11

c) linear search, because we should search names which are always the type of string.

**Exercise2**

bool compare( const Book & s1, const Book & s2 ){

if(s1.publishing\_year<=s2.publishing\_year)

if(s1.publishing\_year=s2.publishing\_year)

if(s1.title<s2.title)

return true;

return false;

}

Int main(){

sort( array, array+SIZE, compare);

}

**Exercise 3**

Vector <int> numbers(2,3); //the elements in numbers are (3 3)

a Numbers.push\_back(4); //now the elements are (3 3 4)

b Numbers.pop\_back(); // now the elements are(3 3)

c Numbers.back(); //return the last data without check if the data exist.

D Numbers.ersae(1)// now the elements are(3)

E operator [] is compatible with the C language. At is our first choice, because it check the boundary, and return out of range

**Exercise 4**

**a** 42 1 2 3 4 5 6 7 8 9 It’s not possible for arrays: v.size() copy=v copy.size()

**b** size means the real size of the vetor while capacity means how large it can be stored.

**Exercise 5**

const int SIZE=50;

int main(){

vector<int> v;

cout<<”Enter integer values: ”<<endl;

for(int i=0;i<SIZE;i==)

cin>>v(i);

sort(v,v+SIZE);

}

**Exercise 6**

Int main(){

Vector<int>numbers;

Int num=0,i=0;

Ifstream fin(“input.txt”);

While(fin>>num)

Numbers(i++)=num;

Int size=numbers.size();

Sort(numbers,numbers+size);

Cout<<”The smallest numer: ”<<number.at(0)<<endl;

Cout<<”The largest number: ”<<number.at(size)<<endl;

Return 0;

}